REMARKS

Claims 1-2, 4, 12-13, and 15-26 are pending. Claims 3, 5-11, 14, and 27 are canceled. Claims 1, 4, 12, 15-16, 19, and 22 are currently amended. The specification is amended.

The Examiner has objected to the specification for failing to provide a proper antecedent basis for the term "actual parameter." Applicant has amended the specification to provide clear antecedent basis for the terms "actual parameter" and "formal parameter." Applicant has attached to this reply an excerpt from a well-known book on compilers: Aho, A., Sethi, R., and Ullman, J., "Compilers: Principles, Techniques and Tools," Addison-Wesley Publishing Company, 1986, p. 390. The excerpt explains the well-known meanings of an "actual parameter" and "formal parameter" as used in the field of computer science. Applicant use of these terms in the specification and claims is consistent with these well-known meanings. The use of "actual" and "formal" adjectives does not constitute new matter because the specification already referred to actual and formal parameters, but simply did not use these well-known terms to describe them.

The Examiner has objected to claim 4 because of a typographical error. Applicant has amended claim 4 to address the Examiner's concern.

The Examiner has rejected claim 15 under 35 U.S.C. § 112 as being indefinite. Applicant has amended claim 15 to address the Examiner's concern.

The Examiner has rejected claims 1-2, 4, and 16-26 under 35 U.S.C. § 103(a) over Kobayashi in view of Murphy and claims 12-13 and 15 under 35 U.S.C. § 103(a) over Kobayashi in view of Murphy and further in view of "MTS." Applicant has amended these claims to clarify the subject matter of the invention.

The pending claims now recite that a new instance of an object or a previously instantiated instance of the object can be tested or exercised. If a new instance is to be

created, then a new instance of the object is created (i.e., instantiating the object) and the new instance is added to an object store. A method of the instance of the object, either the newly created instance or an instance already in the object store, is then invoked to test the object. Claim 1 recites "receiving an indication of whether the object is to be created or already exists in an object store because the object was previously instantiated." Claim 12 recites "receiving an indication of whether to create a new instance of the object or to use an existing instance of the object that is stored in an object store." Claim 16 recites "when the user indicates to test a new instance of the object that does not exist in an object store because of a previous instantiation, instantiating the object and adding the instance of the object to the object store." Claim 22 recites "providing entries that specify an object, [and] whether an instance of the object should be created or chosen from among instances of the object in the object store."

The pending claims also recite that when an invoked method creates another objects that object is added to the object store so that it can be tested or exercised. Claim 1 recites when the invoked method creates another object, adding that other object to the object store so that a user can review the operation of that other object." Claim 12 recites "when the invoked method creates a instance of another object, adding that instance to the object store so that the instance of the other object can be exercised." Claim 16 recites "when it is determined that the invoked method instantiates another object, adding the instance of the other object to the object store so that the instance of the other object can be tested." Claim 22 recites "when the invoked method instantiates another object, adding the instance of the other object to the object store so that the instance of the other object can be tested."

None of the references cited by the Examiner teaches or suggests the combination now recited by applicant's claims. In particular, the claims now recite the combination of allowing the testing of an existing instance of an object that is in an object store or testing of a new instance of an object that is added to the object store and, when an invoked

method of an instance being tested creates an instance of another object, adding that instance to the object store so that that instance can also be tested.

Based on the above amendments and remarks, applicant respectfully requests reconsideration of this application and its early allowance. If the Examiner has any questions or believes a telephone conference would expedite prosecution of this application, the Examiner is encouraged to call the undersigned at (206) 359-8548.

Dated: July 25, 2006

Respectfully submitted,

By Maurice J Pirio

Maurice J. Pirio
Registration No.: 33,273
PERKINS COIE LLP
P.O. Box 1247
Seattle, Washington 98111-1247
(206) 359-8000
(206) 359-7198 (Fax)
Attorneys for Applicant

Compilers

Principles, Techniques, and Tools

ALFRED V. AHO

AT&T Bell Laboratories
Murray Hill, New Jersey

RAVI SETHI

AT&T Bell Laboratories
Murray Hill, New Jersey

JEFFREY D. ULLMAN

Stanford University
Stanford, California

ADDISON-WESLEY PUBLISHING COMPANY
Reading, Massachusetts • Menlo Park, California
Don Mills, Ontario • Wokingham, England • Amsterdam • Sydney
Singapore • Tokyo • Mexico City • Bogotá • Santiago • San Juan

Mark S. Dalton/Publisher
James T. DeWolf/Sponsoring Editor

Bette J. Aaronson/Production Supervisor
Hugh Crawford/Manufacturing Supervisor
Jean Depotan/Cover Design and Illustration
Karen Guardino/Managing Editor

This book is in the Addison-Wesley series in Computer Science
Michael A. Harrison/Consulting Editor

Library of Congress Cataloging in Publication Data

Aho, Alfred V.
Compilers, principles, techniques, and tools.

Bibliography: p.
Includes index.
1. Compiling (Electronic computers) I. Sethi, Ravi. II. Ullman, Jeffrey D., 1942- . III. Title.
QA76.76.C65A37 1985 005.4'53 85-15647
ISBN 0-201-10088-6

Reprinted with corrections March, 1988

Reproduced by Addison-Wesley from camera-ready copy supplied by the authors.

IJK-HA-89

```
(1)  program sort(input, output);
(2)      var a : array [0..10] of integer;

(3)      procedure readarray;
(4)          var i : integer;
(5)          begin
(6)              for i := 1 to 9 do read(a[i])
(7)          end;

(8)      function partition(y, z: integer) : integer;
(9)          var i, j, x, v: integer;
(10)         begin ...
(11)         end;

(12)     procedure quicksort(m, n: integer);
(13)         var i : integer;
(14)         begin
(15)             if ( n > m ) then begin
(16)                 i := partition(m,n);
(17)                 quicksort(m,i-1);
(18)                 quicksort(i+1,n)
(19)             end
(20)         end;

(21)     begin
(22)         a[0] := -9999; a[10] := 9999;
(23)         readarray;
(24)         quicksort(1,9)
(25)     end.
```

Fig. 7.1. A Pascal program for reading and sorting integers.

procedures. A complete program will also be treated as a procedure.

When a procedure name appears within an executable statement, we say that the procedure is *called* at that point. The basic idea is that a procedure call executes the procedure body. The main program in lines 21-25 of Fig. 7.1 calls the procedure `readarray` at line 23 and then calls `quicksort` at line 24. Note that procedure calls can also occur within expressions, as on line 16.

Some of the identifiers appearing in a procedure definition are special, and are called *formal parameters* (or just *formals*) of the procedure. (C calls them "formal arguments" and Fortran calls them "dummy arguments.") The identifiers `m` and `n` on line 12 are formal parameters of `quicksort`. Arguments, known as *actual parameters* (or *actuals*) may be passed to a called procedure; they are substituted for the formals in the body. Methods for setting up the correspondence between actuals and formals are discussed in Section 7.5. Line 18 of Fig. 7.1 is a call of `quicksort` with actual parameters `i+1` and `n`.